

1        EXCEPTION HANDLING COMPILER APPARATUS, PROGRAM, RECORDING  
2                MEDIUM, AND COMPILING METHOD

### 3 FIELD OF THE INVENTION

4 The invention relates to a compiler. More particularly, the invention relates to a compiler  
5 apparatus, compiler program, recording medium, and compiling method that optimize  
6 exception handling.

## 7 BACKGROUND ART

In recent years, programming languages that allow exception handling to be described has become widely used for the purpose of improving maintainability and robustness of programs. When an exception is thrown in a program written in such a programming language, the program shifts its processing from the point in the program where the exception has been thrown to an exception handler that handles the exception. For optimizing the shift of processing to an exception handler, a technology has been utilized that detects types of frequently thrown exceptions and rewrites the instruction that can throw the detected types of exception to a branch instruction leading to an exception handler. See the document, by Takeshi Ogasawara, Hideaki Komatsu, and Toshio Nakatani, "A study of Exception Handling and Its Dynamic Optimization in Java<sup>®</sup> (registered trademark)", Object-Oriented Programming Systems, Languages, and Applications (OOPSLA2001) proceeding, 2001.

20 The technology described above, however, cannot optimize shift of processing to an  
21 exception handler if the types of exception that are to be thrown cannot be determined.  
22 For example, if an exception handler for catching multiple types of exception rethrows an

1 exception it has caught, a compiler apparatus cannot determine how to optimize the  
2 instruction for rethrowing the exception since it cannot decide in advance which type of  
3 exception to catch.

#### 4 **SUMMARY OF THE INVENTION**

5 Therefore, an aspect of the invention is to provide a compiler apparatus, compiler  
6 program, recording medium, and compiling method that can solve the above problem.  
7 The aspect is attained by combinations of features set forth in independent items..  
8 Dependent items define further advantageous and specific examples of the invention.

9 According to an example embodiment of the invention, a compiler apparatus for  
10 optimizing exception handling in a target program as a program to be compiled,  
11 comprises: an exception handler detection section for detecting, from among exception  
12 handlers that catch exceptions thrown in the target program, a multiple-catching  
13 exception handler that catches a plurality of different exceptions and rethrow the caught  
14 exceptions; an exception selection section for selecting a set of exceptions that is to be  
15 shifted to common processing through rethrow of the exceptions by the multiple-catching  
16 exception handler from among a plurality of exceptions caught by the detected  
17 multiple-catching exception handler; and an exception handler throw section for throwing  
18 a clone exception handler that catches the set of exceptions selected by the exception  
19 selection section instead of the multiple-catching exception handler and shifting it to  
20 common processing, a compiler program, a recording medium, and a compiling method  
21 are provided.

22 Although this summary of the invention above does not recite all the necessary features of  
23 the invention, but sub-combinations of these features may be the invention as well.

1     **BRIEF DESCRIPTION OF THE DRAWINGS**

2     For a more complete understanding of the present invention and the advantages thereof,  
3     reference is now made to the following description taken in conjunction with the  
4     accompanying drawings, in which:

5     Fig. 1 is a functional block diagram of a compiler apparatus;

6     Fig. 2 is a flowchart for the compiler apparatus;

7     Fig. 3 shows an example of a target program to be compiled;

8     Fig. 4 shows an example of the target program optimized by the compiler apparatus;

9     Fig. 5(a) shows another example of a target program to be compiled;

10    Fig. 5(b) shows types of exception that are thrown or caught by each instruction in the  
11    target program to be compiled as a function of depth of nesting of function call and  
12    exception catching ranges; and

13    Fig. 6 shows an example of the hardware configuration of the compiler apparatus.

14    **DESCRIPTION OF SYMBOLS**

15           10... compiler apparatus  
16           100... exception handler detection section  
17           110... exception selection section  
18           120... exception handler throw section  
19           500... function  
20           510... exception handler

1           520... exception throwing instruction

2       **DETAILED DESCRIPTION OF THE INVENTION**

3       The present invention provides methods, systems and apparatus to optimize shift of  
4       processing to an exception handler when the types of exception that are to be thrown  
5       cannot be determined. If an exception handler for catching multiple types of exception  
6       rethrows an exception it has caught, a compiler apparatus determines how to optimize the  
7       instruction for rethrowing the exception.

8       In an example embodiment, the present invention provides a compiler apparatus for  
9       optimizing exception handling in a target program as a program to be compiled,  
10      comprising an exception handler detection section for detecting, from among exception  
11      handlers that catch exceptions thrown in the target program, a multiple-catching  
12      exception handler that catches a plurality of different exceptions and rethrow the caught  
13      exceptions; an exception selection section for selecting a set of exceptions that is to be  
14      shifted to common processing through rethrow of the exceptions by the multiple-catching  
15      exception handler from among a plurality of exceptions caught by the detected  
16      multiple-catching exception handler; and an exception handler throw section for throwing  
17      a clone exception handler that catches the set of exceptions selected by the exception  
18      selection section instead of the multiple-catching exception handler and shifting it to  
19      common processing, a compiler program, a recording medium, and a compiling method  
20      are provided.

21      The invention will more particularly be described by means of embodiments in the  
22      following description. Although the embodiments are not intended to limit the invention,  
23      and not all combinations of features set forth in the embodiments are essential for the  
24      solution provided by the present invention.

1 Figure 1 shows a functional block diagram of a compiler apparatus 10. The compiler  
2 apparatus 10 is an apparatus that optimizes exception handling when a target program as  
3 a program to be compiled is compiled, comprising an exception handler detection section  
4 100, an exception selection section 110, and an exception handler throw section 120.  
5 When the exception handler detection section 100 obtains a target program, it detects a  
6 multiple-catching exception handler from among exception handlers that catch exceptions  
7 thrown in the target program for catching a plurality of different exceptions and  
8 rethrowing the caught exceptions, and sends the detection result to the exception selection  
9 section 110.

10 Then, from among the plurality of exceptions caught by the multiple-catching exception  
11 handler that has been detected by the exception handler detection section 100, the  
12 exception selection section 110 selects a set of exceptions that should be shifted to  
13 common processing through rethrow of the exception by the multiple-catching exception  
14 handler and sends the result of the selection to the exception handler throw section 120.  
15 The exception handler throw section 120 then throws a clone exception handler that  
16 catches the set of exceptions selected by the exception selection section 110 instead of the  
17 multiple-catching exception handler and shifts it to common processing. Further, the  
18 exception handler throw section 120 throws a branch instruction for causing a shift to  
19 common processing in the thrown clone exception handler and causes a shift to common  
20 processing with the thrown branch instruction. Subsequently, the exception handler  
21 throw section 120 outputs the program for which the clone exception handler and the  
22 branch instruction were thrown as the result of compilation.

23 In this context, an exception is processing that occurs when processing that does not  
24 conform to a standard predefined for a programming language is performed in executing  
25 a target program, for example. Specifically, an exception occurs when an instruction in a  
26 target program attempts to access an array variable with a subscript that is out of the  
27 range defined for the array variable. Alternatively, an exception may be thrown with an  
28 instruction for autonomously throwing an exception whether rules predefined for the

1 programming language are violated or not. As an example, an exception may be  
2 Exception in Java<sup>®</sup> (registered trademark) language.

3 A target program to be compiled is an intermediate representation thrown from a source  
4 program for efficient optimization and may be byte code in Java<sup>®</sup> (registered trademark)  
5 language, for example. Alternatively, a target program may be RTL (Register Transfer  
6 Language) or quadruple representation. A target program may also be an entire program  
7 to be executed by a user or a module representing a function in the target program. A  
8 module refers to a method, a function, or a procedure, for example.

9 Figure 2 shows a flowchart for the compiler apparatus 10. When the exception handler  
10 detection section 100 obtains a target program to be compiled, it detects, from among  
11 exception handlers for catching exceptions thrown in the target program, a  
12 multiple-catching exception handler that catches a plurality of different exceptions and  
13 rethrows the caught exceptions (S200).

14 Then, from among the plurality of exceptions caught by the multiple-catching exception  
15 handler that has been detected by the exception handler detection section 100, the  
16 exception selection section 110 selects a set of exceptions to be shifted to common  
17 processing through rethrow of the exceptions by the multiple-catching exception handler  
18 (S210). Preferably, the exception selection section 110 selects a set of exceptions further  
19 on condition that the frequency with which they are thrown in the multiple-catching  
20 exception handler is above a predetermined reference frequency. For example, as the  
21 frequency with which exceptions are thrown in the multiple-catching exception handler,  
22 the exception selection section 110 may detect the number of times that any of the set of  
23 exceptions is thrown in the multiple-catching exception handler per the number of  
24 execution of the multiple-catching exception handler, and detect the sets of exceptions  
25 based on the detected number.

26 As a way of detecting the frequency of exception throw, the compiler apparatus 10 may

1 detect it based on information for when a target program that has been compiled once in  
2 another way is actually executed, or may compute an estimated value of the frequency  
3 based on information about the control flow and data flow of the target program.

4 The exception handler throw section 120 then throws a clone exception handler for  
5 catching the set of exception selected by the exception selection section 110 instead of the  
6 multiple-catching exception handler for shifting it to common processing (S220). Then,  
7 the exception handler throw section 120 throws a branch instruction for causing a shift to  
8 common processing in the thrown clone exception handler and causes a shift to common  
9 processing through the thrown branch instruction (S230). The exception handler throw  
10 section 120 also copies processing approximately the same as processing executed in the  
11 multiple-catching exception handler to the clone exception handler so that execution  
12 result of the target program is maintained.

13 In this manner, the compiler apparatus 10 throws a clone exception handler for catching  
14 exceptions instead of a multiple-catching exception handler and shifting them to common  
15 processing, and throws a branch instruction for causing a shift to common processing in  
16 the clone exception handler. This allows the compiler apparatus 10 to omit the rethrow  
17 of caught exceptions and to shift processing to common processing through a mere  
18 branch instruction.

19 The timing with which the series of processing shown in the figure is not limited to  
20 before the execution of the target program to be compiled is started. For example, the  
21 compiler apparatus 10 may throw the clone exception handler during the execution of the  
22 target program that has been compiled once. More specifically, the exception selection  
23 section 110 selects a set of exceptions whose frequency of throw in a multiple-catching  
24 exception handler is above a predefined reference frequency during the execution of the  
25 target program. And, from among the selected sets of exceptions, the exception selection  
26 section 110 selects a set of exception to be shifted to common processing through rethrow  
27 of the exceptions by the multiple-catching exception handler. This allows the exception

1 handler throw section 120 to throw a clone exception handler as necessary during the  
2 execution of the target program. In other words, the compiler apparatus 10 includes the  
3 function of optimizing a target program to be compiled before starting to execute the  
4 program as well as the function of optimizing the target program as appropriate during  
5 the execution of the program, e.g. a function realized as a runtime library.

6 Figure 3 shows an example of target program to be compiled. The braces shown in the  
7 first and thirteenth lines designate the exception catching range for catching exceptions  
8 with the exception handler shown in the thirteenth line. The braces shown in the second  
9 and ninth lines designate the exception catching range for the exception handler shown  
10 from the ninth to twelfth lines. In other words, in the target program depicted, exceptions  
11 thrown in the exception catching range between the second and ninth lines are caught by  
12 the exception handler shown from the ninth to twelfth lines. Further, exceptions thrown  
13 in the exception handler shown between the ninth and twelfth lines are caught by the  
14 exception handler in the thirteenth line. The exception handler in the thirteenth line,  
15 however, catches the exceptions 1 and 2 among exceptions thrown in the exception  
16 handler shown from the ninth to twelfth lines.

17 The instruction in the third and fourth lines is an instruction for throwing the exception 1.  
18 The instruction in the fifth and sixth lines is an instruction for throwing the exception 2.  
19 The instruction in the seventh and eighth lines is an instruction for throwing the exception  
20 3. The instruction in the tenth and eleventh lines is an instruction that rethrows  
21 exceptions caught by the exception handler shown from the ninth to twelfth lines. As  
22 shown, instructions that autonomously throw exceptions specify types of exception. For  
23 example, the instruction in the third and fourth lines throws exceptions of the type  
24 "exception 1". And the exception handlers initiate processing when an exception that  
25 corresponds to the type of exception they specified has been thrown. For example, the  
26 exception handler in the thirteenth line catches the exception 1 and exception 2 which  
27 correspond to the exceptions it specifies among those thrown in its exception detection  
28 range and initiates the processing in the thirteenth line.



1 Upon obtaining the target program in the figure, the exception handler detection section  
2 100 detects the exception handler shown from the ninth to twelfth lines as a  
3 multiple-catching exception handler. Then, from a plurality of exceptions caught by the  
4 multiple-catching exception handler, i.e. the exceptions 1, 2, and 3, the exception  
5 selection section 110 selects a set of exceptions which is to be shifted to common  
6 processing by the multiple-catching exception handler rethrowing the exceptions. For  
7 instance, whether the multiple-catching exception handler throws either of the exception  
8 1 or 2, its processing will be shifted to common processing shown in the thirteenth line,  
9 so the exception selection section 110 selects the exceptions 1 and 2 as a set of exceptions  
10 to be shifted to common processing.

11 In the example in the figure, the multiple-catching exception handler is finally clause in  
12 Java<sup>®</sup> (registered trademark) language and the like for catching any exception thrown in  
13 an exception detection range. Alternatively, the multiple-catching exception handler may  
14 be catch clause that catches multiple exceptions in Java<sup>®</sup> (registered trademark) language  
15 and the like. For example, in Java<sup>®</sup> (registered trademark) language, types of exception  
16 are represented by objects that identify types of exception. Catch clause catches all  
17 objects that are of child class of objects for specified exceptions. That is, the  
18 multiple-catching exception handler may be catch clause that catches exceptions  
19 expressed as objects that are of parent class of multiple objects.

20 That an exception handler catches an exception means that processing of the exception  
21 handler is initiated on condition that the exception is thrown. In particular, that the  
22 exception handler from the ninth to twelfth lines catches the exception 1 means that the  
23 processing of the exception handler from the ninth to twelfth lines is started provided that  
24 the exception 1 is thrown.

25 Figure 4 shows an example of the target program that has been optimized by the compiler  
26 apparatus 10. The exception handler throw section 120 throws a clone exception handler,

1 e.g. the instruction in the ninth and tenth lines, for catching the exceptions 1 and 2 that are  
2 the set selected by the exception selection section 110 and shifting them to common  
3 processing. The exception handler throw section 120 then throws an exception handler in  
4 the eleventh and twelfth lines that catches only the exception 3 instead of the  
5 multiple-catching exception handler shown from the ninth to twelfth lines in Figure 3 so  
6 that the exceptions 1 and 2 which are exceptions of the set selected by the exception  
7 selection section 110 are caught in the clone exception handler. Then, the exception  
8 handler throw section 120 throws a branch instruction in the tenth line for causing a shift  
9 to common processing in the clone exception handler and causes a shift to common  
10 processing with the thrown branch instruction.

11 Thus, the compiler apparatus 10 selects a set of exceptions that are to be shifted to  
12 common processing through rethrow of the exceptions by the multiple-catching exception  
13 handler, and throws a clone exception handler for catching the selected set of exceptions  
14 instead of the multiple-catching exception handler and shifting it to common processing.  
15 In addition, the compiler apparatus 10 throws a branch instruction for causing a shift to  
16 common processing in the clone exception handler. This allows the compiler apparatus  
17 10 to omit processing that is required upon throw of an exception and to shift processing  
18 with a mere branch instruction.

19 Figure 5(a) shows another example of the target program to be compiled. The compiler  
20 apparatus 10 obtains functions 500-1 to N as the target program. The function 500-1  
21 includes an exception handler 510-1 and a call instruction for calling the function 500-2  
22 within the exception detection range for the exception handler 510-1. The function 500-2  
23 includes an exception handler 510-2 and a call instruction for calling the function 500-3  
24 within the exception detection range for the exception handler 510-2. The functions  
25 500-3 to N-1 are similar to the function 500-1 and initiate execution upon being called  
26 sequentially in this order. The function 500-N initiates its execution upon being called by  
27 the function 500-(N-1) and includes the exception handler 510-N and an exception  
28 throwing instruction 520 for throwing an exception in the exception detection range of

1 the exception handler 510-N. A function refers to a portion of a target program that is  
2 predefined by the creator of the target program and may be a method or a procedure.

3 When it obtains the target program, the exception handler detection section 100 detects  
4 two multiple-catching exception handlers, i.e. the exception handler 510-N that is one  
5 multiple-catching exception handler and the exception handler 510-2 that is another  
6 multiple-catching exception handler for catching at least one exception thrown in the  
7 exception handler 510-N. The exception selection section 110 then selects a set of  
8 exceptions from a plurality of exceptions caught by the exception handler 510-N that is to  
9 be shifted to processing in the exception handler 510-2 through rethrow of the exceptions  
10 by the exception handler 510-N and that is then shifted to common processing by  
11 rethrowing the exceptions caught in the exception handler 510-2. For example, when the  
12 exception 1 is thrown in the example in the figure, it will be caught by the exception  
13 handler 510-2 and its processing will be shifted from the exception handler 510-2 to the  
14 exception handler 510-1, so the exception selection section 110 selects the exception 1.

15 In response to that, the exception handler throw section 120 throws each of two clone  
16 exception handlers that correspond to each of the two multiple-catching exception  
17 handlers, and causes each of the two corresponding clone exception handlers to catch the  
18 exception 1 selected by the exception selection section 110 instead of each of the two  
19 multiple-catching exception handlers.

20 Figure 5(b) shows types of exception that are thrown or caught by each instruction in the  
21 target program to be compiled as a function of depth of nesting of function call and  
22 exception catching range. For example, exception instruction 520 is an instruction for  
23 throwing the exception 1, and each of the exception handlers 510-2 and 510-N is an  
24 instruction for catching the exceptions 1, 2, and 3. The exception handler 510-1 is an  
25 instruction for catching the exception 1.

26 The exception handler 510-1 catches outside a function call the exception 1 thrown while

1 processing the function 500-2 that has been called by a function call. The exception  
2 handler 510-2 catches exceptions thrown inside functions called by functions including  
3 the exception handler 510-1. That is, the exception handler 510-2 catches exceptions  
4 thrown between the function 500-2 and the function 500-N. The exception handler  
5 510-N catches exceptions thrown by the exception throwing instruction 520.

6 This means that nesting of function call for the exception handler 510-2 is deeper than  
7 that for the exception handler 510-1. As well, nesting of the exception detection range  
8 for the exception handler 510-2 is deeper than that for the exception handler 510-1.  
9 Similarly, the exception handlers 510-3 to 510-N have deeper and deeper nesting of  
10 function call and exception detection range in this order.

11 Thus, the exception handler 510-N as an example of the multiple-catching exception  
12 handler goes through other exception handlers in shifting processing to the exception  
13 handler 510-1, which is an example of common processing. Therefore, the more number  
14 of other exception handlers processing goes through when it shifts from the exception  
15 handler 510-N to the exception handler 510-1, the more time it takes from throw of an  
16 exception to a shift to common processing.

17 Similarly, the exception handler 510-N goes through the process of recovering from  
18 function calls of the functions 500-N to 500-2 when shifting processing to the exception  
19 handler 510-1. Thus, the deeper the depth of nesting of function call from the exception  
20 handler 510-1 down to the exception handler 510-N is, the more time it takes from throw  
21 of an exception to transition to the common processing.

22 In order to reduce time required for the processing above, the exception selection section  
23 110 preferably selects a set of exceptions further on condition that the number of other  
24 exception handlers which processing goes through during its shift from the exception  
25 handler 510-N to the exception handler 510-1 is more than a predetermined number and  
26 that the depth of nesting of function calls from the exception handler 510-1 down to the

1 exception handler 510-N is more than a predetermined number.

2 That is, from a plurality of exceptions caught by the exception handler 510-N, the  
3 exception selection section 110 selects a combination of exceptions to be shifted to the  
4 exception handler 510-1 through rethrow of the exceptions by the exception handler  
5 510-N, for which the number of other exception handlers through which processing shifts  
6 from the exception handler 510-N to the exception handler 510-1 is more than a  
7 predetermined number and the depth of nesting of function calls from the exception  
8 handler 510-1 down to the exception handler 510-N is more than a predetermined  
9 number.

10 The same applies to a case where more than two multiple-catching exception handler  
11 are detected. For example, the exception selection section 110 selects a set of exceptions  
12 further on condition that the number of other exception handlers through which  
13 processing shifts from a first multiple-catching exception handler via a second  
14 multiple-catching exception handler to a third multiple-catching exception handler is  
15 more than a predetermined number and that the depth of nesting of function calls from  
16 the first multiple-catching exception handler via the second multiple-catching exception  
17 handler down to the third multiple-catching exception handler is more than a  
18 predetermined number.

19 Figure 6 shows an example of the hardware configuration of the compiler apparatus 10.  
20 The compiler apparatus 10 according to the embodiment comprises a CPU peripheral  
21 portion having CPU 1000, RAM 1020, a graphic controller 1075, and a display device  
22 1080 that are interconnected by a host controller 1082; an input/output portion having a  
23 communication interface 1030, hard disk drive 1040, and CD-ROM drive 1060 that are  
24 connected to the host controller 1082 by an input/output controller 1084; and a legacy  
25 input/output portion having ROM 1010, a flexible disk drive 1050, and an input/output  
26 chip 1070 that are connected to the input/output controller 1084.

1 The host controller 1082 connects the RAM 1020 with the CPU 1000 and graphic  
2 controller 1075 that access the RAM 1020 at a high transfer rate. The CPU 1000 operates  
3 based on a compiler program stored in the ROM 1010 and RAM 1020, controlling each  
4 portion. The graphic controller 1075 obtains image data that is thrown by the CPU 1000  
5 and the like in a frame buffer provided in the RAM 1020 and displays it on the display  
6 device 1080. Alternatively, the graphic controller 1075 may internally contain a frame  
7 buffer for storing image data thrown by the CPU 1000 and the like.

8 The input/output controller 1084 connects the host controller 1082 with the  
9 communication interface 1030, the hard disk drive 1040, and the CD-ROM drive 1060  
10 that are relatively high-speed input/output devices. The communication interface 1030  
11 communicates with other devices through a network. The hard disk drive 1040 stores a  
12 compiler program and data to be used by the compiler apparatus 10. The CD-ROM drive  
13 1060 reads the compiler program and data from the CD-ROM 1095 and provides it to the  
14 input/output chip 1070 via the RAM 1020.

15 To the input/output controller 1084, the ROM 1010 and relatively low-speed input/output  
16 devices such as the flexible disk drive 1050 and input/output chip 1070 are connected.  
17 The ROM 1010 stores a boot program to be executed by the CPU 1000 during start-up  
18 time of the compiler apparatus 10 and programs dependent on the hardware of the  
19 compiler apparatus 10. The flexible disk drive 1050 reads the compiler program or data  
20 from the flexible disk 1090 and provides it to the input/output chip 1070 via the RAM  
21 1020. The input/output chip 1070 connects the flexible disk 1090 and various  
22 input/output devices via, for example, a parallel port, serial port, keyboard port and  
23 mouse port.

24 The compiler program to be provided to the compiler apparatus 10 is supplied by a user  
25 as stored in the flexible disk 1090, the CD-ROM 1095, or a recording medium such as an  
26 IC card. The compiler program is read out from the recording medium and installed in  
27 the compiler apparatus 10 via the input/output chip 1070 to be executed in the compiler

1 apparatus 10.

2 The compiler program to be installed in the compiler apparatus 10 and executed includes  
3 an exception handler detection module, an exception selection module, and an exception  
4 handler throw module. Since operations that each module causes the compiler apparatus  
5 10 to perform by operating on it are the same as the operations of corresponding sections  
6 within the compiler apparatus 10 described above with Figures 1 to 5, description about  
7 them is omitted.

8 The program or modules shown above may be stored in an external storage medium. As  
9 the storage medium, optical recording media such as DVD and PD, magneto-optical  
10 recording medium such as MD, tape media, and semiconductor memory such as ID card  
11 can be used, in addition to the flexible disk 1090 and CD-ROM 1095. Also, a storage  
12 device such as hard disk or RAM provided in a server system connected to a dedicated  
13 communication network or the Internet may be used as the recording medium so that the  
14 compiler program is provided to the compiler apparatus 10 over the network.

15 As thus described, the compiler apparatus 10 selects a set of exceptions which is shifted  
16 to common processing through rethrow of the exceptions by a multiple-catching  
17 exception handler and throws a clone exception handler for catching the selected set of  
18 exception instead of the multiple-catching exception handler so as to shift it to common  
19 processing. Further, the compiler apparatus 10 throws a branch instruction that causes a  
20 shift to common processing in the clone exception handler. This allows the compiler  
21 apparatus 10 to omit processing that is required upon throw of an exception and to shift  
22 processing with a mere branch instruction. In addition, since it can extend a compilation  
23 scope, which is an area in a target program which other optimization such as partial  
24 redundancy elimination and the like optimizes, the compiler apparatus 10 can increase the  
25 efficiency of optimization.

26 Although the invention has been described with its embodiment, the technical scope of

1 the invention is not limited to the scope described in the embodiment above. Rather,  
2 various modification and improvement may be made to the embodiment. It is obvious  
3 from the items below that a form with such modification or improvement may also be  
4 encompassed within the technical scope of the invention.

5 According to the embodiment set forth above, the compiler apparatus, compiler program,  
6 recording medium, and compiling method can be realized described in the following  
7 items.

8 (Item 1)

9 A compiler apparatus for optimizing exception handling in a target program as a  
10 program to be compiled, comprising:

11 an exception handler detection section for detecting, from exception handlers that  
12 catch exceptions thrown in the target program, a multiple-catching exception handler that  
13 catches a plurality of different exceptions and rethrow the caught exceptions;

14 an exception selection section for selecting a set of exceptions that are to be shifted  
15 to common processing through rethrow of the exception by the multiple-catching  
16 exception handler from among the plurality of exceptions caught by the detected  
17 multiple-catching exception handler; and

18 an exception handler throw section for throwing a clone exception handler that  
19 catches the set of exceptions selected by the exception selection section instead of the  
20 multiple-catching exception handler and shifting it to common processing.

21 (Item 2)

22 The compiler apparatus according to Item 1, wherein the exception handler throw  
23 section throws a branch instruction for causing a shift to the common processing in the  
24 clone exception handler and causes a shift to the common processing with the branch  
25 instruction thrown.

26 (Item 3)



1           The compiler apparatus according to Item 1, wherein the exception selection section  
2 selects a set of exceptions whose frequency of throw in the multiple-catching exception  
3 handler is more than a predetermined reference frequency and which is shifted to the  
4 common processing.

5 (Item 4)

6           The compiler apparatus according to Item 3, wherein, as a frequency with which  
7 exceptions are thrown in the multiple-catching exception handler, the exception selection  
8 section detects the number of times that any of the set of exceptions is thrown in the  
9 multiple-catching exception handler per the number of execution of the multiple-catching  
10 exception handler.

11 (Item 5)

12           The compiler apparatus according to Item 1, wherein  
13 the common processing includes catching exceptions that are thrown while  
14 processing a function that has been called with a function call outside the function call;  
15 the multiple-catching exception handler catches exceptions inside the function call;  
16 and  
17 the exception selection section selects the set of exceptions further on condition that  
18 depth of nesting of function call from the common processing down to the  
19 multiple-catching exception handler is more than a predetermined number.

20 (Item 6)

21           The compiler apparatus according to Item 1, wherein the exception selection section  
22 selects the set of exceptions further on condition that the number of other exception  
23 handlers through which processing shifts from the multiple-catching exception handler to  
24 the common processing is more than a predetermined number.

25 (Item 7)

26           The compiler apparatus according to Item 1, wherein

1           the common processing includes catching exceptions that are thrown while  
2   processing a function that has been called with a function call outside the function call;  
3           the multiple-catching exception handler catches exceptions inside the function call;  
4   and  
5           the exception selection section selects the set of exceptions further based on depth  
6   of nesting of function call from the common processing down to the multiple-catching  
7   exception handler and the number of other exception handlers through which processing  
8   shifts from the multiple-catching exception handler to the common processing.

9   (Item 8)

10          The compiler apparatus according to Item 1, wherein  
11          the exception handler detection section detects two multiple-catching exception  
12   handlers: one multiple-catching exception handler and another multiple-catching  
13   exception handler for catching at least one exception thrown in the one multiple-catching  
14   exception handler;  
15          the exception selection section selects a set of exceptions to be shifted to the  
16   common processing by rethrowing an exception caught in the another multiple-catching  
17   exception handler from among a plurality of exceptions caught by the one  
18   multiple-catching exception handler; and  
19          the exception handler throw section throws each of two clone exception handlers  
20   that correspond to each of the two multiple-catching exception handler and causes each of  
21   the corresponding two clone exception handlers to catch the set of exceptions selected by  
22   the exception selection section instead of each of the two multiple-catching exception  
23   handler.

24   (Item 9)

25          A compiler program for causing a computer to function as a compiler apparatus that  
26   optimizes exception handling in a target program as a program to be compiled, the  
27   compiler program causing the computer to function as:

28          an exception handler detection section for detecting, from exception handlers that

1 catch exceptions thrown in the target program, a multiple-catching exception handler that  
2 catches a plurality of different exceptions and rethrow the caught exceptions;  
3 an exception selection section for selecting a set of exceptions that are to be shifted  
4 to common processing through rethrow of the exceptions by the multiple-catching  
5 exception handler from among the plurality of exceptions caught by the detected  
6 multiple-catching exception handler; and  
7 an exception handler throw section for throwing a clone exception handler that  
8 catches the set of exceptions selected by the exception selection section instead of the  
9 multiple-catching exception handler and shifting it to the common processing.

10 (Item 10)

11 The compiler program according to Item 9, wherein  
12 the common processing includes catching exceptions that are thrown while  
13 processing a function that has been called with a function call outside the function call;  
14 the multiple-catching exception handler catches exceptions inside the function call;  
15 and  
16 the exception selection section selects the set of exceptions further based on depth  
17 of nesting of function call from the common processing down to the multiple-catching  
18 exception handler and the number of other exception handlers through which processing  
19 shifts from the multiple-catching exception handler to the common processing.

20 (Item 11)

21 The compiler program according to Item 9, wherein  
22 the exception handler detection section detects two multiple-catching exception  
23 handlers: one multiple-catching exception handler and another multiple-catching  
24 exception handler for catching at least one exception thrown in the one multiple-catching  
25 exception handler;  
26 the exception selection section selects a set of exceptions to be shifted to the  
27 common processing by rethrowing an exception caught in the another multiple-catching  
28 exception handler from among a plurality of exceptions caught by the one

1 multiple-catching exception handler; and  
2 the exception handler throw section throws each of two clone exception handlers  
3 that correspond to each of the two multiple-catching exception handler and causes each of  
4 the corresponding two clone exception handlers to catch the set of exceptions selected by  
5 the exception selection section instead of each of the two multiple-catching exception  
6 handler.

7 (Item 12)

8 A recording medium having the compiler program according to any of Items 9 to 11  
9 recorded thereon.

10 (Item 13)

11 A compiling method for causing a computer to operate as a compiler apparatus that  
12 optimizes exception handling in a target program as a program to be compiled,  
13 comprising:

14 an exception handler detection step of detecting, from exception handlers that catch  
15 exceptions thrown in the target program, a multiple-catching exception handler that  
16 catches a plurality of different exceptions and rethrow the caught exceptions;

17 an exception selection step of selecting a set of exceptions that are to be shifted to  
18 common processing through rethrow of the exceptions by the multiple-catching exception  
19 handler from among a plurality of exceptions caught by the detected multiple-catching  
20 exception handler; and

21 an exception throw step of throwing a clone exception handler that catches the set  
22 of exceptions selected by the exception selection section instead of the multiple-catching  
23 exception handler and shifting it to the common processing.

24 As is obvious from the description above, exception handling can be optimized according  
25 to the invention. Variations described for the present invention can be realized in any  
26 combination desirable for each particular application. Thus particular limitations, and/or  
27 embodiment enhancements described herein, which may have particular advantages to the

1 particular application need not be used for all applications. Also, not all limitations need  
2 be implemented in methods, systems and/or apparatus including one or more concepts of  
3 the present invention.

4 The present invention can be realized in hardware, software, or a combination of  
5 hardware and software. A visualization tool according to the present invention can be  
6 realized in a centralized fashion in one computer system, or in a distributed fashion where  
7 different elements are spread across several interconnected computer systems. Any kind  
8 of computer system - or other apparatus adapted for carrying out the methods and/or  
9 functions described herein - is suitable. A typical combination of hardware and software  
10 could be a general purpose computer system with a computer program that, when being  
11 loaded and executed, controls the computer system such that it carries out the methods  
12 described herein. The present invention can also be embedded in a computer program  
13 product, which comprises all the features enabling the implementation of the methods  
14 described herein, and which - when loaded in a computer system - is able to carry out  
15 these methods.

16 Computer program means or computer program in the present context include any  
17 expression, in any language, code or notation, of a set of instructions intended to cause a  
18 system having an information processing capability to perform a particular function  
19 either directly or after conversion to another language, code or notation, and/or  
20 reproduction in a different material form.

21 Thus the invention includes an article of manufacture which comprises a computer usable  
22 medium having computer readable program code means embodied therein for causing a  
23 function described above. The computer readable program code means in the article of  
24 manufacture comprises computer readable program code means for causing a computer to  
25 effect the steps of a method of this invention. Similarly, the present invention may be  
26 implemented as a computer program product comprising a computer usable medium  
27 having computer readable program code means embodied therein for causing a a function

1 described above. The computer readable program code means in the computer program  
2 product comprising computer readable program code means for causing a computer to  
3 effect one or more functions of this invention. Furthermore, the present invention may be  
4 implemented as a program storage device readable by machine, tangibly embodying a  
5 program of instructions executable by the machine to perform method steps for causing  
6 one or more functions of this invention.

7 It is noted that the foregoing has outlined some of the more pertinent objects and  
8 embodiments of the present invention. This invention may be used for many  
9 applications. Thus, although the description is made for particular arrangements and  
10 methods, the intent and concept of the invention is suitable and applicable to other  
11 arrangements and applications. It will be clear to those skilled in the art that  
12 modifications to the disclosed embodiments can be effected without departing from the  
13 spirit and scope of the invention. The described embodiments ought to be construed to be  
14 merely illustrative of some of the more prominent features and applications of the  
15 invention. Other beneficial results can be realized by applying the disclosed invention in  
16 a different manner or modifying the invention in ways known to those familiar with the  
17 art.